

# A Methodology for Oracle Selection of Monitors and Knobs for Configuring an HPC System running a Flood Management Application

Panagiota Nikolaou  
University of Cyprus

Yiannakis Sazeides  
University of Cyprus

Antoni Portero  
IT4Innovations, National  
Supercomputing Center

Radim Vavrik  
IT4Innovations, National  
Supercomputing Center

Vit Vondrak  
IT4Innovations, National  
Supercomputing Center

## ABSTRACT

This paper defines a methodology for the oracle selection of the monitors and knobs to use to configure an HPC system running a scientific application while satisfying the application's requirements and not violating any system constraints. This methodology relies on a heuristic correlation analysis between requirements, monitors and knobs to determine the minimum subset of monitors to observe and knobs to explore to determine the optimal system configuration for the HPC application.

At the end of this analysis, we reduce an 11-dimensional space to a 3-dimensional space for monitors and a 6-dimensional space to a 3-dimensional space for knobs. This reduction shows the potential and highlights the need for a realistic methodology to help identify such minimum set of monitors and knobs.

## 1. INTRODUCTION

High Performance Computing (HPC) infrastructures run a diverse set of applications with different Quality-of-Service (QoS) requirements [10]. These requirements come in various forms, such as operational power, performance, energy, cost and availability. Naturally, HPC systems need to be configured in a way to satisfy those application requirements [17]. To configure a system, all the different system hardware and software knobs are set at specific settings, with many remaining at a default setting, and then, while the application runs, various monitors are observed to determine whether the various requirements are satisfied. Unsurprisingly, the configuration space is extremely large and such configuration search efforts are in practice adhoc and non-optimal.

One way to reduce the configuration search dimensionality and complexity, it is to reduce the requirements, monitors and knobs that need to be satisfied, observed and explored, respectively. Reduction of a problem's dimensionality is not a new problem for computing system analysis [13, 12, 14, 9]. Such reduction, typically, relies on some form of statistical correlation, for example, principal component analysis [19, 6].

In this work we explore the potential benefits, from

a yet to be determined, realistic methodology that can identify the minimum set of monitors and knobs to use for configuring an HPC system that runs a specific application while satisfying the application requirements. To accomplish this, we use a correlation methodology that relies on data obtained from a detail exploration of a configuration space. Even though, a detail exploration is what a practical search methodology should avoid, the analysis in this paper, is useful to assess the potential benefits that can come from a future methodology that reduces the search space.

Specifically, for this investigation we use Floreon+ application, a flood prediction and management application, running on an HPC platform. The analysis considers data obtained using eleven system monitors when exploring many settings for six knobs. This analysis is performed for individual and combinations of the following requirements: performance, power, availability, energy and cost. The results reveal that the configuration space can be reduced considerably. Additionally, the results show non-obvious correlations between requirements, monitors and knobs. This motivates the need for further research to determine a practical configuration space reduction methodology for HPC systems.

The rest of the paper is organized as follows: Section 2 covers the background related to the Floreon+ [23] application and HPC organization. Section 3 describes the characterization and experimental framework and correlation analysis, and, Section 4 presents and discusses experimental results. This paper concludes in Section 5.

## 2. BACKGROUND

### 2.1 HPC Application (Floreon+)

In this work, we use Floreon+ (FLOod REcognition On the Net) [23], an HPC application with high QoS requirements. Floreon+ is an online system for monitoring, modeling, prediction and support disaster flood management [20]. The system focuses on acquiring and analyzing relevant data in near-real time. The data are used to provide short-term flood prediction by running hydrologic simulations.

The main processes of Floreon+ application are organized as follows:

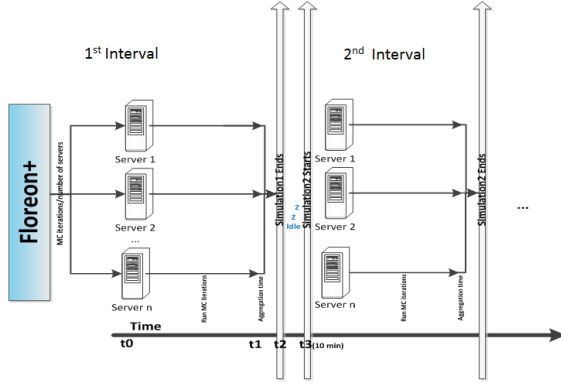


Figure 1: Floreon+ normal operation

1. Get information about actual river and reservoir situation.
2. Rainfall-runoff (RR) modeling: simulation of surface runoff.
3. Hydrodynamic (HD) modeling: flood lake simulations, flood maps, simulations of water elevation and water velocity, a real-time hydrological model for flood prediction, water quality analysis, etc.
4. Erosion modeling: simulation of water erosion.
5. Collection and archiving of flood data that can be used to estimate the magnitude of a flood based on historical evidence.

In this work we are investigating the uncertainty of the Rainfall-runoff (RR) modeling which is the most computationally intensive part [20, 23].

The application framework for the uncertainty of RR model provides an environment for running multiple simulations every repetition, when new data arrives on an HPC system. The uncertainty contains information about how accurate is the solution that RR model provides. RR model is a dynamic mathematical model, which transforms rainfall to flow at the catchment outlet.

The uncertainty is computed as Monte Carlo samples. The Monte Carlo method gives a straightforward way of massive parallelism by increasing the number of random values working concurrently to obtain numerical results. Previous experiments [20] exhibit a good scalability of the Monte Carlo method in an HPC cluster with 64 nodes of 16 cores each. Consequently, the proposed methodology in this paper may be appropriated in any other HPC framework where Monte Carlo method is employed.

Figure 1 shows the normal operation of Floreon+. As Figure 1 shows, a batch of Monte Carlo iterations is running in a number of nodes (Server 1- Server n) in such a way that application's QoS requirements are satisfied. Each interval indicates the execution of a different simulation. For example, the 1<sup>st</sup> interval refers to the 1<sup>st</sup> simulation. After the execution of all the Monte Carlo iterations, the results send to a master server for processing. The total simulation time includes the execution time of Monte Carlo iterations and the time needed to process the results. When a simulation ends, the servers remain idle for a set of period. The duration of this period is determined by the availability of the new batch of data. Under normal operation (fault-free) the

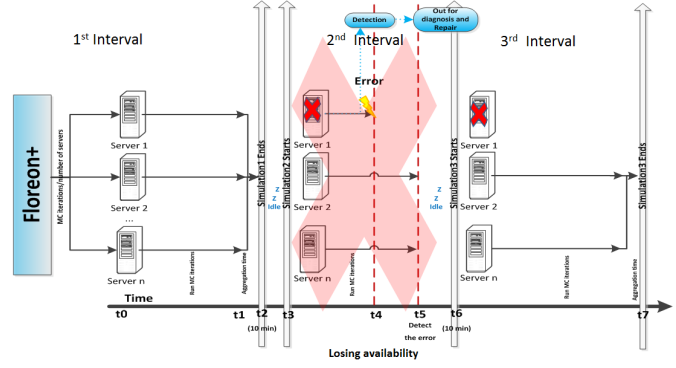


Figure 2: Floreon+ operation with faults

simulation always finishes within the time constraint. There are some cases where a fault on a component can delay the execution of the simulation, as shown in Figure 2. These cases can be categorized in the following:

1. Delay the execution of the simulation but still the simulation finishes within the time constraint. The availability of the system does not decrease.
2. Delay the execution and violate the timing constraint with the same number of servers. Thus the results of this simulation are useless and the availability of the system decreases.
3. Delay the execution and violate the timing constraint with less servers. In this case the faulty server needs to be taken offline until it is repaired or replaced. In this case the results are lost and the availability decreases.

Figure 2 illustrates the last case where the server needs to be taken offline until it is repaired or replaced. As shown in the figure, the number of servers in the 3<sup>rd</sup> interval is decreased and the job is assigned to the remaining servers until the faulty server is repaired or replaced.

Next we present the various requirements for this application.

### 2.1.1 Application Requirements

Floreon+ has two running operation modes, the standard operation mode and the emergency operation mode. Both have different requirements. Standard operation mode is the default operation of the system. In this operation the weather is favorable and the flood warning level is below of the critical threshold.

On the other hand, on the emergency operation mode the water in the rivers rises due to continuous rain or free-flowing streams that are created due to heavy rainfall on small areas. During this operation mode much more accurate and frequent simulation computations are needed and the results should be provided as soon as possible. In this work we focus on the emergency operation which has tighter timing requirements.

### 2.1.2 Availability and Quality of Service requirements

The reliability and availability target of Floreon+ running on emergency operation is accomplished through a combination of hardware/software mechanisms and policies. This also aims at satisfying the QoS requirements even in the presence of errors and offline servers.

**Table 1: Floreon+ requirements**

Performance	simulation $\leq 10$ minutes
Accuracy	$\geq 10^4$ MC iterations
Availability	$\geq 0.99$
Power	$\leq 81$ Watts
Energy	$\leq 48600$ Joules

These mechanisms typically rely on hardware and software **monitors** and **knobs**.

In general, when a server fails and if its repair time is expected to be long, the system software migrates the failed job to another server. The failure of the server is detected by a hardware or software monitor. The migration is possible because for QoS and availability reasons. HPC systems are over-provisioned with spares for dealing with errors and offline servers. Server over-provisioning is determined by the availability of a system. The less available system, the more servers needed [22]

Because of its significance, emergency operation requires responsiveness in 10 minutes for each simulation. Also it must provide high levels of availability, two nines (0.99), which may require over-provisioning with extra servers to deal with various hardware failures.

Floreon+ and other offline services can run together (collocated) to improve utilization [21, 31]. Specifically, when Floreon+ satisfies the QoS requirements without using all the available cores in a server, the remaining cores can run other services. This must be done without affecting the QoS of Floreon+ and violating its requirements. Since we are going to explore the emergency operation, Floreon+ is running in isolation, utilizing all the available resources without any other service concurrently runs on the same server.

### 2.1.3 Accuracy on Monte Carlo Iterations

It is of utmost importance that the results are as precise as they can. The precision of the simulated results is based on the number of Monte Carlo iterations [24]. It has been shown in [24] that the number of iterations has to be in the order of  $10^4$  to  $10^5$  to obtain a satisfying precision. In this work, we assume 20000 Monte Carlo iterations that has to be computed before the deadline (i.e. ten minutes, since new input data arrives from weather stations) as the baseline configuration.

Table 1 summarizes the specific values for the requirements of the Floreon+ application.

## 2.2 Available Monitors and Knobs in HPC Systems

A large number of monitors and knobs exist in HPC systems. Monitors enable the observation of physical, micro-architectural, and operating system phenomena that can assess the status of a system as well as the progress towards completing a task. On the other hand, knobs enable the proactive or reactive control of various phenomena.

Monitors and knobs in real systems can be categorized into the following categories depending on what metric they influence: performance, power, temperature and reliability.

Table 2 shows the requirements, monitors and knobs that are going to be explored in this work. This subset is by no means comprehensive and future work should consider a larger set.

Monitors that are going to be investigated are: ex-

**Table 2: Application requirements, Monitors and Knobs**

Requirements	Monitors	Knobs
Performance	Execution Time	DVFS
Power	IPC	SMT
Energy	DRAM Power	DRAM Protection
Availability	CPU Power	Turbo Mode
Cost	Peak Power	Prefetchers
	CPU Temperature	Redundancy
	MPKI	
	Server MTBF	
	System MTBF	
	Capex expenses	
	Opex expenses	

**Table 3: Server Configuration**

Number of CPUs	2
CPU	Intel Xeon E5-2640 v3
Number of cores per CPU	8
Number of threads per core	2
Channels per CPU	4
DIMMs/channel	2
DIMM capacity	16GB

ecution time, Instructions per Cycle (IPC), Misses per kilo Instructions (MPKI), DRAM, CPU and peak power and CPU temperature. Also, Mean Time Between Failures (MTBF) per server and for the whole system is used through analytical models and Failures In Time (FIT) rates [25]. Finally, capital (CAPEX) and operational (OPEX) expenses are going to be estimated based on publicly available info, e.g. list prices, and runtime measurements. CAPEX expenses include infrastructure, server and networking equipment costs, whereas OPEX expenses include power and maintenance costs.

Table 2, also shows the different knobs that we are going to experiment with. As the table shows, we use Simultaneous Multi-Threading (SMT) [28], Dynamic Voltage and Frequency Scaling (DVFS) [30], data prefetchers [27] and Intel’s Turbo Mode [16]. Also, this work provides results with and without redundant cores. Redundant cores are used to improve the reliability of the system by migrating the running thread of a faulty core to a spare [15]. For the redundancy scenario it is assumed that half of the cores remain idle to provide higher availability. On the other hand, in the scenario without redundancy all the available physical resources are utilized. Furthermore, this study explores the implication of using two different DRAM Protection Techniques (No Protection or ChipkillDC).

DRAM is protected from errors by using extra devices per DIMM to store Error Correction Code (ECC) bits. Modern processors usually support Chipkill with 16 ECC bits to protect 128 data bits that are interleaved across two DIMMs placed in two channels [3, 8, 4]. This is referred as ChipkillDC or Lockstep where it can correct all the errors in a single device and detects all the errors in two devices [3]. Chipkill can waste bandwidth, hurt performance and increase energy consumption [7, 18, 29]. On the other hand No Protection

**Table 4: Values of knobs**

Knobs	Value
DVFS	1.2, 1.7, 2.2, 2.6 (GHz)
SMT	Disable, Enable
DRAM Protection	No Protection, ChikillDC
Turbo Mode	Disable, Enable
Prefetchers	Disable, Enable
Redundancy	Disable, Enable

does not provide any protection on DRAM.

### 3. EXPERIMENTAL FRAMEWORK AND CORRELATION ANALYSIS

#### 3.1 Experimental Framework

For evaluation, we use an HPC cluster with dual socket Intel Xeon E5-2640 v3 system configuration, as shown in Table 3. We run each experiment 5 times, and each time we observed execution time, instructions per cycle (IPC), misses per kilo instructions (MPKI), CPU, DRAM power and CPU temperature. The results presented are calculated by removing minimum and maximum values and calculating the average.

Prefetchers and DRAM protection techniques changed by accessing BIOS, through a BIOS Serial Command Console interface (CLI) [5].

Our evaluation used *Floreon+*, an HPC application with a dataset of 44KB (Tiny dataset). This is a representative dataset size for the application purposes because it uses five days observations to provide predictions for the next two days. All the power numbers are collected using the Likwid-powermeter [1] which allows monitoring the power consumption of CPU and DRAM at any given time. The results are used to calculate total power and peak power numbers.

To track CPU temperature, we use lm-sensors [2]. To estimate Server MTBF and System MTBF monitoring values, as well as, availability values we use different analytical models based on binomial probabilities.

Also, to estimate CAPEX, OPEX expenses as well as total cost we use COST-ET and AMPRA tools proposed in [11, 22].

For each experiment, an initial population of 2 server modules is used. The number of servers can change depending on availability and performance requirements.

For a baseline configuration, we select the one that is currently used to run this application and includes the following values for each parameter: SMT:OFF, Frequency: 2.6 GHz, DRAM Protection: No Protection, Turbo Mode: Enable, Redundancy: 0 (No), Prefetchers: ON.

The data used for correlation analysis are obtained by exploring the 128 combinations of knobs presented in Table 4. For each configuration combination the eleven monitor values are recorded. These eleven values are then used to determine the values of five metrics each corresponding to a specific requirement.

The values of the requirements for a given configuration are obtained as follows: performance from execution time, power from cpu and dram power, energy product of execution time and power, availability from server MTBF, server MTTR and number of servers, and cost using monitors for power, performance, availability using the COST-ET tool [11]. The total data set used for this analysis is, therefore, a 128 x 11 x 5 data ma-

trix, where 128 are the combinations of knobs, 11 are the monitors values and 5 are the requirements.

In this Section we describe the methodology used for correlation analysis to reduce the number of requirements, monitors and knobs used to configure an HPC system. The data that drive this analysis are obtained as discussed in Section 3.1. The value obtained for each requirement for a given experiment, are normalized to give each requirement equal weight. The normalization is done by computing the mean and standard deviation for each requirement and then by subtracting the mean and dividing the standard deviation by each value of requirements as in [13]. A configuration is considered to be successful if it satisfies each individual application requirement. To rank successful configurations we determine an overall score for a configuration by multiplying each normalized requirement value with an equal weight. In this case, since we have 5 requirements, we multiply each value with 0.2. The correlation analysis is done using the R statistical language [26].

##### 3.1.1 Heuristic Correlation Analysis

This analysis explores the correlation between requirements, monitors and knobs. The methodology used is as follows:

1. For a given requirement, we compute the correlation coefficient (using Pearson correlation analysis) with all the other requirements. For each pair  $(x_i, x_j)$  of requirements  $i$  and  $j$ , where  $i \neq j$ , that exhibit significant correlation coefficient (above a 90% threshold <sup>1</sup>), we check which of the two requirements can be removed. The requirement that shows smaller correlation coefficient with all the other requirements is removed from the list. This process is iterated over all remaining requirements. The same process is repeated for the monitors.
2. For the remaining requirements and monitors, we compute the correlation coefficient between each requirement and all remaining monitors and select the monitor with the highest correlation.
3. For all the remaining monitors and all the available knobs we compute the correlation coefficient between them and knobs that have a correlation coefficient above a 40% threshold <sup>1</sup> are kept.

This correlation analysis aims to reduce the number of configurations that need to be explored to determine the configuration that provides the highest satisfaction of the requirements according to the ranking in Section 3.1. Specifically, the analysis returns a subset of the monitors and knobs. All possible configuration combinations are then evaluated for the selected knobs. For the knobs that are not selected we used the baseline configuration values (we have confirmed that it does not really matter which value you use for them). Afterwards, the selected configurations are sorted according to the selected monitor(s) value(s) (if there is more than one monitor, equal weighting is used to combine them). The top ranked configuration using the selected knobs and monitors is then compared with the configuration that considers all. Their difference is measured as the maximum negative % difference for any of the requirements (if it is 0 for all it means it matches the best possible configuration).

<sup>1</sup>The specific thresholds are picked on empirical analysis not shown here

**Table 5: Correlation between Monitors and Requirements**

Requirements	Monitors
Performance	Execution time
Availability	Capex cost
Power	CPU power
Cost	Capex cost
All	Execution time, CPU power, Capex cost

## 4. RESULTS

### 4.1 Results of the Heuristic Correlation Analysis

Our analysis reveals that Energy is strictly correlated with performance and thus we removed Energy from the list of requirements and the presented results.

Table 5 shows the results of correlation analysis between requirements and monitors.

As we can see from the Table, Performance requirement can be monitored by Execution Time. Availability can be, monitored by Capex expenses. This correlation is not obvious because Capex cost has an indirect relation with Availability. The correlation of Capex expenses and availability comes from the extra servers that are needed for over-provisioning. The more servers needed for over-provisioning, the more capex expenses are incurred. Moreover, the more servers are needed for over-provisioning, the less the System MTBF due to the higher probability of errors. So, this leads to a smaller availability for this system. As the Table, also, shows, CPU power is the appropriate metric for the total power. This happens because Floreon+ is not a memory intensive application so it has a negligible impact on DRAM power. For the total cost and overall combination of requirements, capex is the correlated monitoring value.

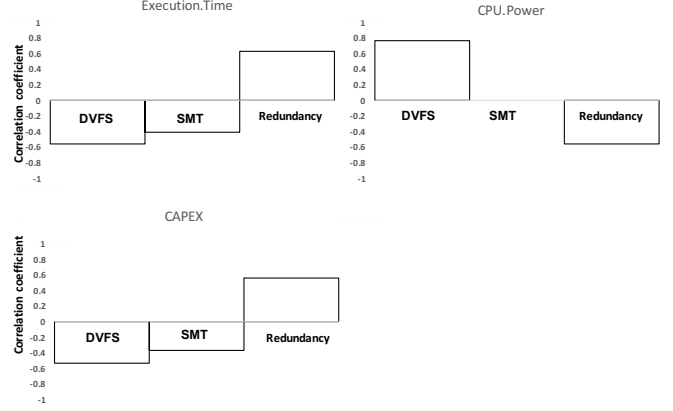
Finally, Figure 3 shows, the correlation between knobs and the monitors. X-axis presents the remaining list of knobs for each monitor, whereas the y-axis presents the correlation coefficient between knobs and monitors. The correlation coefficient ranges from -1 to +1. A value closer to +1 means that this knob has almost a linear relation with the monitor. A value closer to -1 means that this knob has an inverse relation with the monitor. A value closer to 0 means that there is no correlation between the knob and monitor.

As we can see from the Figure, DVFS, SMT and Redundancy are the selected knobs. On the other hand, DRAM Protection, Turbo Mode and Data prefetching can be reduced from the search space. This is due to the low cache and low memory pressure of the Floreon+ application (L3MPKI: 0.0034, L2MPKI: 0.013).

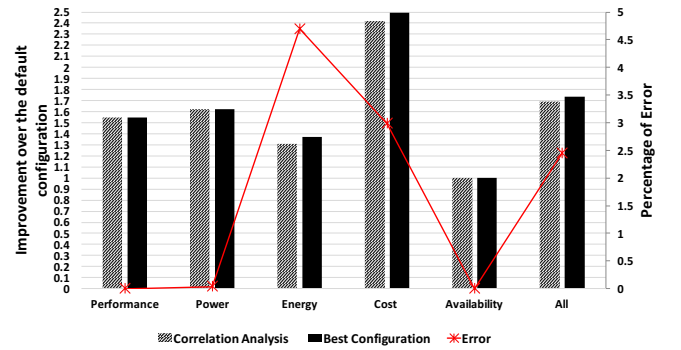
### 4.2 Validation of Heuristic Methodology

Figure 4 validates the heuristic analysis by evaluating the best configuration for each requirement using the subset of monitors and knobs revealed from the correlation analysis.

In this graph, we show the normalized with the default configuration, improvement for each individual requirement and all requirements between the best configuration using the selected monitors and knobs and the best configuration using all monitors and knobs. As can be seen from Figure 4, all the requirements can be improved by changing the system configuration. For ex-



**Figure 3: Correlation analysis presenting the correlation coefficient between Monitors and Knobs**



**Figure 4: Improvement between the default configuration, correlation analysis predicted configuration and the best configuration for different requirements**

ample, performance can be improved by 1.5x and cost by 2.4x related to the default configuration. Also, the combination of all the requirements can be improved by 1.7x. Moreover, the Figure shows that the configurations based on the correlation analysis are very close to the results with the best configuration. We measure the error between the two configurations and we found a maximum value of 5%.

## 5. CONCLUSIONS AND FUTURE WORK

This paper describes an oracle methodology that investigates combinations of different knobs and the effects they have on the various monitored values. This aims to explore the potential benefits, from a yet to be determined, realistic methodology that can reduce the number of monitors and knobs that configure an HPC system that runs a specific application while satisfying the application requirements. To this end, this analysis reduces an eleven space of system monitors to three and a six system space knobs to three.

Findings of this work motivates future research. One direction is to find a realistic methodology probably based on correlation analysis that will be generic for

many applications. Another direction is to minimize the data sample used.

## 6. ACKNOWLEDGMENT

The research leading to this paper is supported by the “Harpa, Project No 612069” of the European Commission 7th RTD Framework Program, the “Uniserver, Project No 688540” of the European Community’s H2020 Program, the Czech Republic Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project, IT4Innovations National Supercomputing Center – LM2015070 and the University of Cyprus.

## 7. REFERENCES

- [1] likwid-powermeter: Tool for accessing RAPL counters on Intel processor. <https://github.com/RRZE-HPC/likwid/wiki/Likwid-Powermeter>.
- [2] lm-sensors. <http://www.lm-sensors.org/wiki/man/sensors-detect>.
- [3] BIOS and Kernel Developers Guide (BKDG) for AMD Family 10h. April 2010.
- [4] BIOS and Kernel Developers Guide (BKDG) for AMD Family 15h. February 2014.
- [5] Hp rom-based setup utility user guide. February HP TR, 2014.
- [6] H. Abdi and L. J. Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.
- [7] J. H. Ahn, N. P. Jouppi, C. Kozyrakis, J. Leverich, and R. S. Schreiber. Future scaling of processor-memory interfaces. In *Conference on High Performance Computing Networking, Storage and Analysis*, pages 42:1–42:12, 2009.
- [8] S. Ankireddi and T. Chen. Configuring and using DDR3 memory with HP ProLiant Gen8 Servers, Best Practice Guidelines for ProLiant servers with Intel Xeon processors. February 2014.
- [9] M. Annavaram, R. Rakvic, M. Polito, J.-Y. Bouguet, R. A. Hankins, and B. Davies. The fuzzy correlation between code and performance predictability. In *Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture*, pages 93–104. IEEE Computer Society, 2004.
- [10] E. Brewer. Lessons from giant-scale services. *Internet Computing*, pages 46–55, 2001.
- [11] D. Hardy, M. Kleanthous, I. Sideris, A. Saidi, E. Ozer, and Y. Sazeides. An analytical framework for estimating tco and exploring data center design space. In *International Symposium on Performance Analysis of Systems and Software*, pages 54–63, 2013.
- [12] K. Hoste and L. Eeckhout. Comparing benchmarks using key microarchitecture-independent characteristics. In *2006 IEEE International Symposium on Workload Characterization*, pages 83–92. IEEE, 2006.
- [13] K. Hoste and L. Eeckhout. Microarchitecture-independent workload characterization. *IEEE Micro*, 27(3):63–72, 2007.
- [14] K. Hoste, A. Phansalkar, L. Eeckhout, A. Georges, L. K. John, and K. De Bosschere. Performance prediction based on inherent program similarity. In *Proceedings of the 15th international conference on Parallel architectures and compilation techniques*, pages 114–122. ACM, 2006.
- [15] L. Huang and Q. Xu. Characterizing the lifetime reliability of manycore processors with core-level redundancy. In *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 680–685. IEEE, 2010.
- [16] Intel. Intel Turbo Boost Technology in Intel Core microarchitecture (Nehalem) based processors. White paper. Technical report, November 2008.
- [17] A. Iordache, E. Buyukkaya, and G. Pierre. Heterogeneous resource selection for arbitrary hpc applications in the cloud. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 108–123. Springer, 2015.
- [18] X. Jian, H. Duwe, J. Sartori, V. Sridharan, and R. Kumar. Low-power, low-storage-overhead chipkill correct via multi-line error correction. In *International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 24:1–24:12, 2013.
- [19] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.
- [20] S. Kuchar, M. Golasowski, R. Vavrik, M. Podhoranyi, B. Sir, and J. Martinovic. Using high performance computing for online flood monitoring and prediction. *International Journal of Environmental, Ecological, Geological and Geophysical Engineering*, 9(5):267–272, 2015.
- [21] J. Mars, L. Tang, R. Hundt, K. Skadron, and M. L. Soffa. Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations. In *44th Annual International Symposium on Microarchitecture*, pages 248–259, 2011.
- [22] P. Nikolaou, Y. Sazeides, L. Ndreu, and M. Kleanthous. Modeling the implications of dram failures and protection techniques on datacenter tco. In *Proceedings of the 48th International Symposium on Microarchitecture, MICRO-48*, pages 572–584, New York, NY, USA, 2015. ACM.
- [23] A. Portero, Š. Kuchař, R. Vavřík, M. Golasowski, and V. Vondrá. System and application scenarios for disaster management processes, the rainfall-runoff model case study. In *IFIP International Conference on Computer Information Systems and Industrial Management*, pages 315–326. Springer, 2014.
- [24] A. Portero, R. Vavrik, S. Kuchar, M. Golasowski, V. Vondrak, S. Libutti, G. Massari, W. Fornaciari, and I. e Bioengegneria. Flood prediction model simulation with heterogeneous trade-offs in high performance computing framework. In *29th EUROPEAN Conference on Modelling and Simulation ECMS*, 2015.
- [25] V. Sridharan, J. Stearley, N. DeBardeleben, S. Blanchard, and S. Gurumurthi. Feng shui of supercomputer memory: Positional effects in dram and sram faults. In *International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 22:1–22:11, 2013.
- [26] R. C. Team et al. R: A language and environment for statistical computing. 2013.
- [27] J. M. Tendler, J. S. Dodson, J. Fields, H. Le, and B. Sinharoy. Power4 system microarchitecture. *IBM Journal of Research and Development*, 46(1):5–25, 2002.
- [28] D. M. Tullsen, S. J. Eggers, and H. M. Levy. Simultaneous multithreading: Maximizing on-chip parallelism. *SIGARCH Comput. Archit. News*, 23(2):392–403, May 1995.
- [29] A. N. Udipi, N. Muralimanohar, R. Balsubramonian, A. Davis, and N. P. Jouppi. Lot-ecc: Localized and tiered reliability mechanisms for commodity memory systems. In *39th Annual International Symposium on Computer Architecture*, pages 285–296, 2012.
- [30] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced cpu energy. In *Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation, OSDI ’94*, Berkeley, CA, USA, 1994. USENIX Association.
- [31] H. Yang, A. Breslow, J. Mars, and L. Tang. Bubble-flux: Precise online qos management for increased utilization in warehouse scale computers. In *40th Annual International Symposium on Computer Architecture*, pages 607–618, 2013.